Paper describing the data:

<https://pubmed.ncbi.nlm.nih.gov/30899034/>

Full list of samples:

<https://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP132189&o=acc_s%3Aa&s=SRR6671775,SRR6671776,SRR6671777,SRR6671757,SRR6671758,SRR6671759>

reference genome:

<https://www.ncbi.nlm.nih.gov/assembly/GCA_015534855.1>

Getting started

Check conda and python are installed on your machine

conda –-version

if conda is not found, download with:

wget https://repo.anaconda.com/miniconda/Miniconda3-py39\_4.12.0-Linux-x86\_64.sh

sha256sum Miniconda3-py39\_4.12.0-Linux-x86\_64.sh

you should see a long hash code if conda has been downloaded correctly

bash Miniconda3-py39\_4.12.0-Linux-x86\_64.sh

follow the instructions to install, close and reopen your command line when finished

conda list

a list of packages will appear if installed correctly

Create a conda environment to store your RNAseq analysis tools and activate it.

conda create -n RNAseq

y when prompted

conda activate RNAseq

Download reads from NCBI sequence read archive.

conda install sra-tools -c bioconda

y when prompted

mkdir RNAseq\_training

cd RNAseq\_training

mkdir raw\_reads

cd raw\_reads

for k in SRR6671757 SRR6671758 SRR6671759 SRR6671775 SRR6671776 SRR6671777; do fasterq-dump $k; echo $k; done

gzip \*.fastq

Perform quality control on raw reads using fastQC and multiQC

<https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/02_assessing_quality.html>

<https://multiqc.info/docs/>

Install fastQC and multiQC to your conda environment

conda install fastqc multiqc -c bioconda

y when prompted

Navigate to the RNA-seq directory in your command line using the cd (change directory) command

cd {your path}/RNAseq-training

Make a new subdirectory within the RNA-seq directory to store your fastqc reports

mkdir qc-reports

Making sure command line is pointing to the folder with the reads in, a for loop can be used to run fastqc on all FASTQ files

for filename in \*.fastq;

do fastqc $filename -o qc-reports/;

echo $filename;

done

The variable ‘filename’ is defined as any files with .fastq.gz in their name, and FastQC is called using ‘filename’ as its input. FastQC then iterates through all applicable files generating a report for each one. echo $filename prints the filename to the command line when each iteration is completed

Alternatively, the \* wildcard can be used to perform matching within the expression without the need for iterating through the directory which can be slow.

fastqc \*fastq.gz -o qc-reports/

Run multiQC to aggregate all fastQC reports

multiqc .

Trimming Reads with Trimmomatic

<http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf>

Install trimmomatic to your conda environment

conda install trimmomatic -c bioconda

y when prompted

Run trimmomatic on reads, replace {sample} with one of the sample names

trimmomatic PE {sample}\_1.fastq {sample}\_2.fastq.gz {sample}\_1-trimmed-paired.fastq {sample}\_1\_trimmed-unpaired.fastq {sample}\_2-trimmed-paired.fastq {sample}\_2-trimmed-unpaired.fastq ILLUMINACLIP:NexteraPE-PE.fa:2:30:10 LEADING:3 TRAILING:3 MINLEN:36 SLIDINGWINDOW:4:15

trimmomatic PE

{sample}\_1.fastq

{sample}\_2.fastq.gz

{sample}\_1-trimmed-paired.fastq

{sample}\_1\_trimmed-unpaired.fastq

{sample}\_2-trimmed-paired.fastq

{sample}\_2-trimmed-unpaired.fastq

ILLUMINACLIP:NexteraPE-PE.fa:2:30:10 LEADING:3 TRAILING:3 MINLEN:36 SLIDINGWINDOW:4:15

Running each file individually is boring and inefficient, for loops can be constructed to recursively run a command for each applicable file in the folder

for infile in \*\_1.fastq.gz

do base=$(basename ${infile} \_1.fastq.gz)

trimmomatic PE ${infile} ${base}\_2.fastq.gz ${base}\_1-trimmed-paired.fastq.gz ${base}\_1-trimmed-unpaired.fastq.gz ${base}\_2-trimmed-paired.fastq.gz ${base}\_2-trimmed-unpaired.fastq.gz ILLUMINACLIP:NexteraPE-PE.fa:2:30:10 LEADING:3 TRAILING:3 MINLEN:36 SLIDINGWINDOW:4:15 2> ${base}.log

echo ${infile}

done

Breaking this down:

for infile in \*\_1.fastq.gz;

Defines the arbitrarily named variable ‘infile’ as any file in the current directory with R1.fastq.gz in its name,

do base=$(basename ${infile} \_1.fastq.gz);

Defines a new arbitrarily named variable ‘base’ which uses the basename command to remove the R1.fastq.gz suffix from the ‘infile’ variable, thus leaving only the sample name prefix for the command to loop through

trimmomatic PE ${infile} ${base}\_2.fastq.gz ${base}\_1-trimmed-paired.fastq.gz ${base}\_1-trimmed-unpaired.fastq.gz ${base}\_2-trimmed-paired.fastq.gz ${base}\_2-trimmed-unpaired.fastq.gz LEADING:3 TRAILING:3 MINLEN:36 SLIDINGWINDOW:4:15

Calls Trimmomatic in the normal way, note {base} is used instead of the sample name

2> ${base}.log;

Writes the output which is normally printed on the command line to a log file

echo ${infile};

Prints the name of the sample in the command line when each loop finishes

done

Ends command

The raw and trimmed reads can then be moved into new directories using the mv (move) command

mkdir raw-reads trimmed-paired trimmed-unpaired

mv \*unpaired.fastq.gz trimmed-unpaired/

mv \*paired.fastq.gz trimmed-paired/

mv \*.fastq.gz raw-reads/

Aligning Trimmed Reads to Reference Genome with Bowtie2

(Alignment is a computationally intensive process and will take an hour or so, so this is just for completeness)

<https://bowtie-bio.sourceforge.net/bowtie2/manual.shtml>

Install bowtie2 to your conda environment

conda install bowtie2 -c bioconda

y when prompted

Make a new directory for the alignment files and navigate to it

mkdir read-alignment

cd read-alignment

Download a reference genome to the bowtie2-alignment directory and and build a bowtie index

curl <https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/015/534/855/GCA_015534855.1_ASM1553485v1/GCA_015534855.1_ASM1553485v1_cds_from_genomic.fna.gz> -o ecoli\_K12\_ref.fna.gz

gunzip ecoli\_K12\_ref.fna

bowtie2-build ecoli\_K12\_ref.fna ecoli\_K12\_ref

Navigate to the directory with the trimmed paired reads and run the alignment

cd ..

cd trimmed\_paired

for infile in \*1-trimmed-paired.fastq.gz; do base=$(basename ${infile} 1-trimmed-paired.fastq.gz); bowtie2 –x {PATH}/read-alignment/ecoli\_K12\_ref -1 ${infile} -2 ${base}2-trimmed-paired.fastq.gz -S {PATH}/read-alignment/ecoli\_K12\_ref read-alignment/${base}.sam 2> {PATH}/${base}.log; echo ${base}; done

Alignment-free Transcript Quantification with Salmon

Salmon can align and quantify trimmed reads without the need to use a dedicated alignment tool

<https://combine-lab.github.io/salmon/getting_started/>

Install Salmon to your conda environment

conda install salmon -c bioconda

y when prompted

mkdir transcript\_quant

curl https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/015/534/855/GCA\_015534855.1\_ASM1553485v1/GCA\_015534855.1\_ASM1553485v1\_cds\_from\_genomic.fna.gz -o ecoli\_K12\_ref.fna.gz

salmon index -t ecoli\_K12\_ref.fna -i /transcript\_quant/ecoli\_K12\_ref

for infile in \*\_1-trimmed-paired.fastq.gz; do base=$(basename ${infile} \_1-trimmed-paired.fastq.gz); salmon quant --libType A -i ecoli\_K12\_ref -o transcript\_quant/${base} -p 8 –-validateMappings -1 ${infile} -2 ${base}\_2-trimmed-paired.fastq.gz; echo ${infile}; done

You now have a quantified transcriptome for each sample. Open the .sf files in a text editor to inspect the calculated expression of each gene.

OR if alignment has been performed

Navigate to your read-alignment directory and run salmon to quantify transcripts aligned to the reference genome

cd ..

cd read-alignment

for filename in \*.sam; do salmon quant -t ecoli\_K12\_ref.fna --libType A -a $filename -o ${filename}\_quant; echo $filename; done

Calculating Differential Expression Between Groups with Deseq2

<https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html>

The degree a gene is upregulated/downregulated can be determined based on the proportion of reads in the test condition relative to the control. This, along with the data visualisation, will be done in R.